山形大学紀要（工学）第17巻　第1号　昭和57年1月
Bull. of Yamagata Univ., Eng., Vol. 17, No. 1, Jan. 1982

SAMOS シミュレータおよび最小命令数構成

中津山幹男・水沼　充・杉本 皆子
西塚　典生・長橋　宏

工学部　電子工学科

（昭和56年8月27日受理）

1. はじめに

最近の電子計算機の発展は目覚しいものがあり，ますます大型化し，メモリーの容量もメガバイトからギガバイトの時代になり従来考えられなかった大規模なプログラムの開発が可能となってきた。一方，計算速度も数MIPS*から百MIPSと飛躍的に発展してきた。計算速度の要求から，回路は増々小型化してきている。ハードウエアの進展にともない，ソフトウエア特にシステム・プログラムは巨大化しつつあり，システム・プログラム記述言語も機械語やアセンブリ言語ではプログラムの生産性が悪く，PL/Iや設計専用言語に頼らない限りシステム・プログラムが書けなくなってきた。

プログラム作製の手段として機械語を用いなければならないような機会は急速に減少しつつあるが，電子計算機の機能を知るためには，ハードウエアの構成図，マイクロプログラムの流れ図とともに機械語は電子計算機の内容を知るための恰好の手段である。機械語の難点は，その命令形式が2進，8進，16進等の数値で表わされることにあり，到底，全てを記憶することは難かしい。また，入出力の周辺機器を完全に動かすためには，ステータスのチェック，割込機能の理解など，学習すべきことが多く，入門者にとっては大変難解な言語に映ろう。より学習を容易にするためには，機械語と一対一の対応のある低レベルのアセンブリ言語[1]を用いることがあるが，ここでも入出力の完全な記述は学習が難しく，また記号番地などはプログラミングを容易にするが，メモリとの対応に現実感が薄れていく嫌いがある。

SAMOS[2]は非常に簡略化した機械語であり，記憶しやすい形に整備されており，入出力も思い切った簡単化を図っており，入門者にとって非常に理解しやすい言語である。本論文では，ミニコンピュータTOSBAC−40上にSAMOSシミュレータを作製し，さらに学習しやすいようにデバック機能を追加した。また，SAMOSの最少命令数について検討し，興味ある結果が得られたので報告する。

2. SAMOSの概略

SAMOSは架空の電子計算機システムの言語ではあるが，そのシステムは現実のシステ

*MIPS : Million Instructions Per Second

ムに似せてあり，他のシステムへの応用は容易である。

2.1 SAMOSの電子計算機の構成

SAMOS用の電子計算機システムは現実のシステムとほぼ一致しており，Fig. 1 に示す通りである。プログラムおよびデータは入力装置より入力されて記憶装置に格納される。プログラムは0番地よりスタートし，HALT 命令に到達したときに計算が終了し，そこで制御はモニターに移り，次のジョブの処理を行なう。

SAMOSを理解するためには，各種のレジスタの動作を知らなければならない。その動

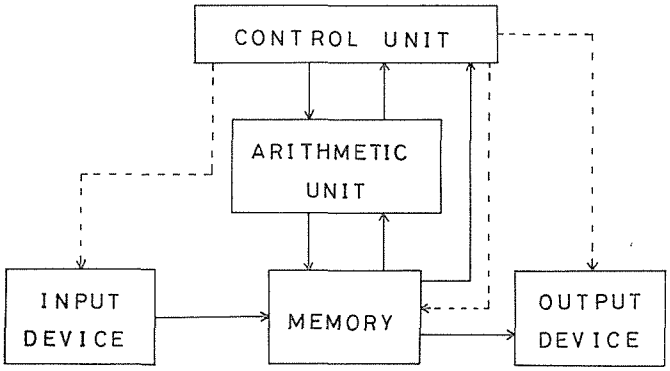


Fig. 1　Hardware organization of a computer system.

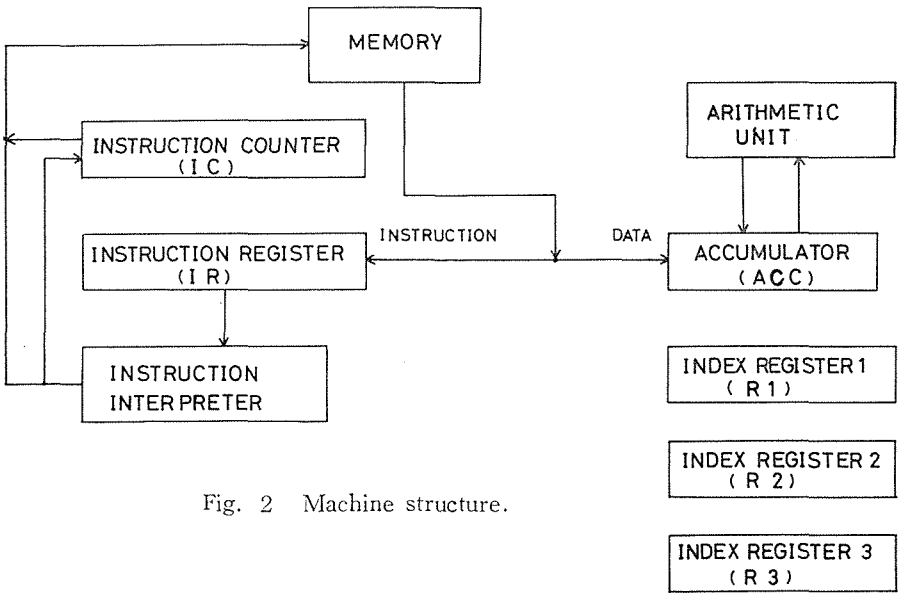


Fig. 2　Machine structure.

作を概念的に画くとFig 2のようになる。命令カウンタ（IC）の指示する番地より命令を受取り，命令レジスタ（IR）へ転送する。命令レジスタは操作レジスタ（OR）と番地レジスタ（AR）より成り，操作部を解読して，必要な操作指令を演算装置に送り，さらに番地の処理を行なう。データの演算では，番地部レジスタが示す番地よりデータを読み取りアキュムレータへ転送する。演算は，必ずアキュムレータを経由して行なうのが，このシステムの特長である。最新のシステムでは汎用のレジスタを16こ持ち，アキュムレータ，インデックス・レジスタとして自由自在に使用しているが，機械語の習得にはアキュムレータのみの方が理解しやすい。命令が実行されるとＩＣは自動的に１だけ増し，また分岐命令のときは条件により番地部がＩＣに格納される。

2.2 命令体系

SAMOSの命令は，入門者の便宜のためできるだけ少くしてあるが，より上級者の練習のために若干の命令が追加されている。基本的な命令はTable 1の通りである。表中ADRは番地，（ADR）は番地の内容を表わす。また（ACC），（IC），（Ri）などは，それぞれACC，IC，Riの内容を示す。表の他に浮動小数点に関する命令があるが，ここでは略

Table 1 SAMOS instructions

OPERATION CODE	MEANING
LDA (LOAD)	(ACC) ← (ADR)
STO (STORE)	(ADR) ← (ACC)
ADD (ADD)	(ACC) ← (ACC)+(ADR)
SUB (SUBTRACT)	(ACC) ← (ACC)−(ADR)
MPY (MULTIPLY)	(ACC) ← (ACC)*(ADR)
DIV (DIVIDE)	(ACC) ← (ACC)/(ADR)
HLT (HALT)	STOP
BRU (BRANCH UNCONDITIONALLY)	(IC) ←ADR
BMI (BRANCH CONDITIONALLY)	IF (ACC) ＜0 THEN (IC) ←ADR
RWD (READ A WORD)	(ADR) ← (INPUT DEVICE)
WWD (WRITE A WORD)	(OUTPUT DEVICE) ← (ADR)
SHL (SHIFT LEFT)	(ACC) ← (ACC)$\times 10^{ADR}$
SHR (SHIFT RIGHT)	(ACC) ← (ACC)$\times 10^{-ADR}$
LIi (LOAD INDEX)	(Ri) ← (ADR)$_{8\sim11}$ i=1,2,3
SIf (STORE INDEX)	(ADR)$_{8\sim11}$← (Ri) i=1,2,3
TIi (TEST INDEX)	(Ri) ← (Ri)−1 IF Ri=0 THEN (IC) ←ADR i=1,2,3

す。条件付き分岐命令はただ一つしかない。通常の機械語では（ACC）が負の時のみでなく，正の時，零の時などに分岐する命令を備えているのが普通であるが，BMI だけでも十分であることを Fig. 3 の流れ図で示す。Fig. 3（a）は正の時の分岐，（b）は零の時の分岐する流れ図である。図で ADR は分岐先の番地を示す。Fig. 3 は，より簡単な流れ図で表わすことが可能であるが，ここでは，原理的な図を示しておく。

他に論理演算の AND，OR，NOT，EXCLUSIVELY，OR などの命令をもつ機械語が多いが，これらは基本的な四則演算で代用することができるので論理演算命令がなくても良い。SAMOS ではインデックス・レジスタが多く，これによりプログラム作製がかなり容易になっている。

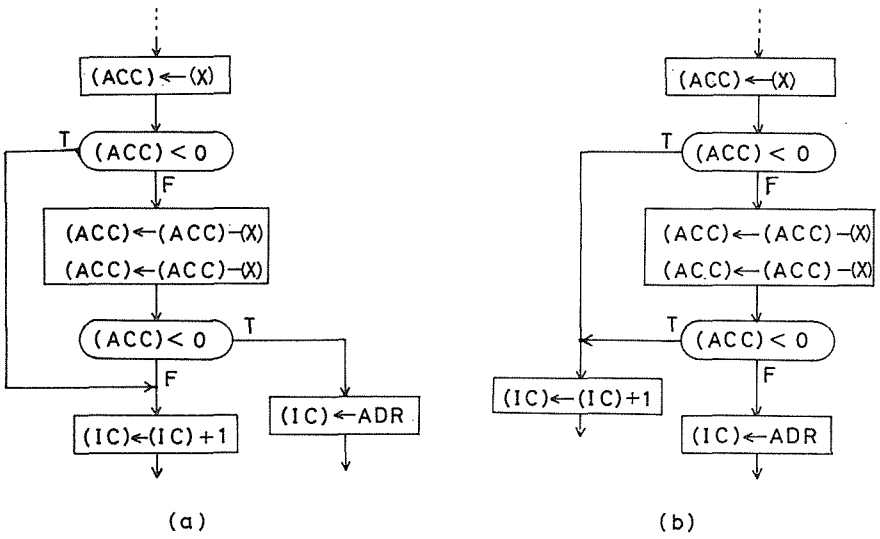


Fig. 3　(a)　Branch on plus.　　(b)　Branch on ZERO.

3．最小命令構成

命令数が，どこまで簡略化できるかは，機械演算が始まって以来の問題であり，当然ハードウエアのシステムに大きく依存しており，汎用の電子計算機では一般性のある解答は得ることができない。電子計算機の究極的な形は Turing Machine や Post Machine に帰するので，この Machine での研究がなされている。[3)]

SAMOS は幸い架空の電子計算機の機械語であり，ハードウエアからなかば独立した言語であるので，最小命令構成の検討が比較的容易である。以下省略できる命令について検討をしてみる。

3.1　乗除算命令の省略

乗算は省略可能であることは容易に理解できる。いま

z←X*Y

について流れ図を書くと Fig. 4 のようになる。簡単のため X≧0，Y≧0 とおく。Fig. 4 の箱は一つの命令に対応しここでは LDA，STO，BMI，ADD のみで乗算が構成できる。

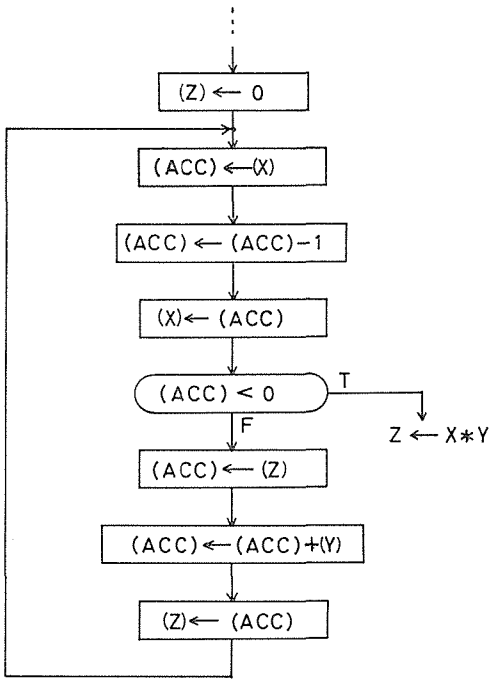


Fig. 4 Multiplication.

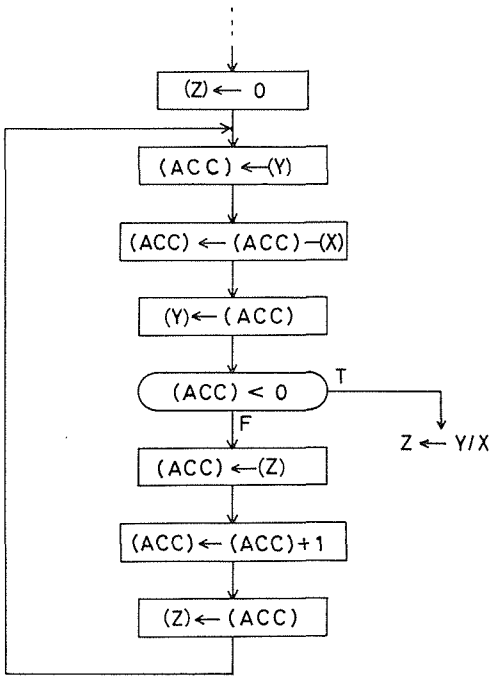


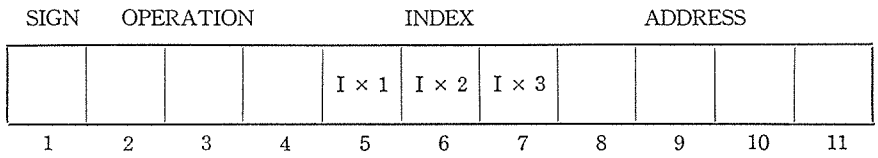
Fig. 5 Division

明らかに，もっと高速の演算のための流れ図が可能であるが，ここでは原理図のみを示す。

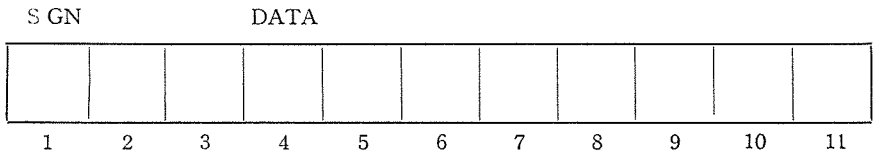
除算の流れ図も，Fig. 5 にほぼ同様の形で示される。やはり X＞ 0，Y≧0とおく。この制限は一般性を損なうものではない。XとYの符号が異なるときに，答が負となり，同符号の時，答が正になるようにすれば良い。除算ではＳＵＢ命令がさらに必要となる。

3.2　シフト命令の省略

SAMOSの命令形式，データ形式は Fig. 6 の通りである。Ｉ×１，Ｉ×２，Ｉ×３は，０のときインデックス・レジスタの指定がされず，１のとき該当のインデックス・レジスタが指定される。シフトにより符号の内容は変らず，右シフトでは，左より零が入り，左シフトでは右より零が入ってくる。したがって左シフトでは，１桁シフトするごとに10を乗じ，右シフトでは１桁シフトするごとに10で除することに相当する。



(a)　INSTRUCTION FORMAT



(b)　DATA FORMAT

Fig. 6　Instruction format and data format.

3.3　インデックス・レジスタの省略

インデックス・レジスタは番地修飾，カウンタなどに用いられる。Fig. 7 (a) は番地修飾のプログラム例であり，110番地から101番地までのデータを90番地から81番地までに移している。Fig. 7 (b) では最終的には同じ結果となるが101番地から110番地までのデータを81番地から90番地までに移している。ただし，Fig. 7 (b) のプログラムは，実行するたびにプログラムの内容が書き換えられることになるので，良いプログラムとは言えない。Fig. 7 (a) でインデックス・レジスタはカウンタの役目も果しており，LI 1，TI 1がカウンタとしての機能を持っている。Fig. 7 (b) では56番地から60番地までがカウンタの働きをしている。

3.4　無条件分岐命令の省略

Fig. 7 (a) で51，52番地はBRU命令の動作をしており，BMIで代用できることがわか

```
        LI1   000  0069
  ⋮      ⋮
40      LDA   100  0100
41      STO   100  0080
  ⋮      ⋮
50      TI1   000  0053
51      LDA   000  0071
52      BMI   000  0040
53      ·····
  ⋮
69      +000  000  0010
70      +000  000  0001
71      −000  000  0001
```

(a)

```
40      LDA   000  0101
41      STO   000  0081
  ⋮      ⋮
50      LDA   000  0040
51      ADD   000  0070
52      STO   000  0040
53      LDA   000  0041
54      ADD   000  0070
55      STO   000  0041
56      LDA   000  0072
57      ADD   000  0070
58      STO   000  0072
59      SUB   000  0069
60      BMI   000  0040
  ⋮      ⋮
69      +000  000  0010
70      +000  000  0001
71      −000  000  0001
72      +000  000  0000
```

(b)

Fig. 7 A program (a) using an index register can be rewritten into a program (b) using no index register.

る。

3.5 HLT 命令の省略

HLT の考えられる機能に2種あり，一つは電子計算機を文字通り止めてしまう場合と，他はモニタに制御を移す場合である。前者は特殊なレジスタにデータを入力する必要があり，HLT がこの命令を兼ねる場合は一応省略できない。しかし，この場合でも記憶装置外の番地の割当てや，正しくない命令の格納などにより強制的に止めることで代用することが可能である。またモニタに制御を移す場合は分岐命令で書き換えることができる。以上の検討により HLT は省略可能である。

3.6 RWD, WWD 命令の省略

RWD 命令によるデータ入力は，データの記憶，番地の確保，すなわちリテラルにより代用できる。WWD は特定の番地の内容を何らかの手段で紙に印刷することになるので，このままでは他の命令で代用することは不可能である。しかし，WWD を何らかの方法でユーザに値を知らせるものと再定義すれば WWD は不用になる。たとえば，電子計算機のコンソールには通常ランプで各種レジスタやアキュムレータの内容を指示している。アキュムレータの内容が BCD コードで示されている場合は WWD は不用となる。通常，これに似たアキュムレータの内容の指示の方法が行われているので，RWD とともに WWD は不用となる。

3.7　最小命令数

以上を総合して，必要な命令は LDA，STO，ADD，SUB，BMI の5種になる。機械語によっては，LDA を CLEAR AND ADD という命令で置き換えることがある。アキュムレータの内容を零にして加える，つまり LDA に他ならない。LDA を実行する前に必らずアキュムレータを零にすることにすれば，STO，SUB，ADD で置き換えることができる。したがって最小命令は STO，ADD，SUB，BMI の4種類になる。

ここで留意すべき点は，最小命令により一般にプログラムの長さが大きくなり，それだけ実行時間が増大することである。最も大きな影響を与えるのが乗除算であり実行時間は少く見積っても数百倍も遅くなることが問題点である。したがって4種類の命令は実用的には殆んど問題にならないが，冗長でない機械語命令を設計するための基礎になると考えられる。

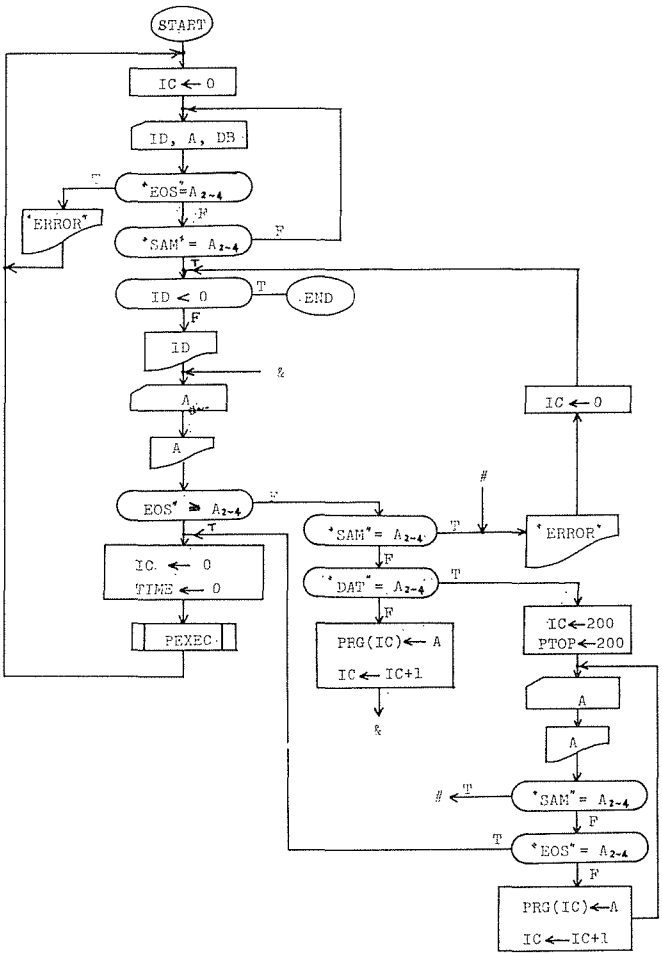


Fig. 8　Main flowchart of SAMOS simulator

4. SAMOSインタプリタ

SAMOSをインタプリタ形式で作製することとした。コンパイラ形式の方が高速であるが，エラー発生時にTOSBAC－40ではシステムが破壊されてしまうこと，その2次的影響で統計的資料が得られなくなること，連続処理ができにくいことなどの理由からインタプリタ形式とした。インタプリタ形式の良い点は，デバックが容易であること，エラーがすべてシステム内で完全にかつ安全に処理できることである。

4.1 インタプリタ

LISP[4]やBASIC[5]では元来インタプリタ形式で書かれているのであるが，最近では高速性を重視するためコンパイラ[6][7]形式に移行しているようである。

インタプリタ形式の長所はLISPなどのように自己拡張が可能で，システムがだんだん

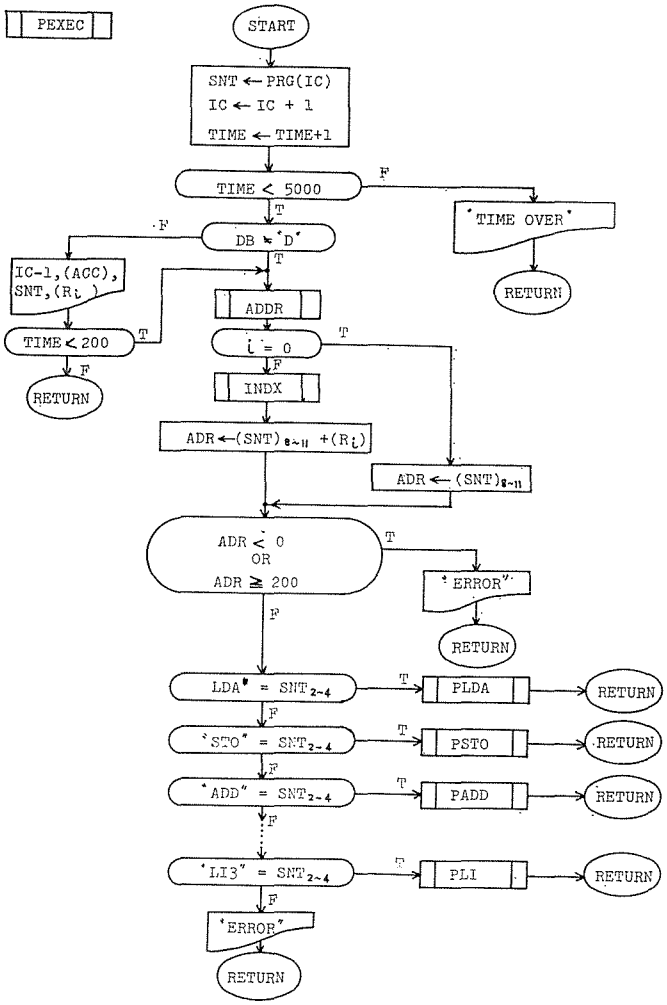


Fig. 9 Flowchart of PEXEC

成長することができる。もちろん拡張中に不用な部分ができると削除したり詰め込んだりする手順が必要になってくる。また BASIC のように現在実行中のステートメントを調べることが可能となる。短所は，ステートメントをその都度解釈しては実行しなければならないので，コンパイラ形式に比して 100 倍程度遅いことである。

SAMOS をインタプリタ形式とした理由は上述した通りであるが，他に初心者の教育用として考えると，プログラムは一般に小さく，実行時間も小さいので，インタプリタの欠点が気にならないことである。またインタプリタ形式により容易に時間超過のチェックが可能である。

SAMOS インタプリタは文字の扱いが不便という欠点はあるが FORTRAN で記述した。FORTRAN は他の電算機で一般に使われ，移植も容易である。

4.2 インタプリタの流れ図

SAMOS インタプリタの主な流れ図は Fig. 8 の通りである。A は大きさ11の配列で，命令およびデータの読み取りに用いられる。ID はユーザの番号である。PRG は0番地から199番地までの命令およびリテラルを表わす。PRG の大きさは若干小さめであるが，機械語の練習には十分であり素数を求めるプログラムも十分 PRG の中におさまる。データ・カードは200番地から249番地までに格納される。RWD 命令で直接カードリーダから読み込む方法が自然であるが，データ・カードの内容が明らかになること，エラーの処理が容易であることから，データをあらかじめ記憶する方式を採用した。DB はデバックを表わし，トレースができるようになっている。また，ジョブの入力時には命令のエラー

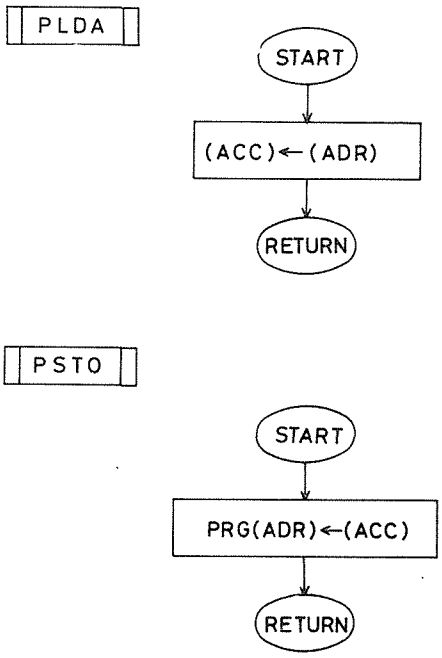


Fig. 10　Flowcharts of PLDA and PSTO.

・チェックは行わない。一般に機械語では命令とデータの区別ができないという原則に従っているのである。

PEXEC の流れ図を Fig. 9 に示す。命令カウンタ IC の示す番地の内容を SNT に移し，もし DB＝"D" ならばデバックのトレース機能が要求されているので，番地，アキュムレータ ACC の内容，SNT およびインデックスレジスタの内容を印刷する。ただしデバックの場合は TIME は 200 で打切ることとした。普通の演算では TIME が 5000 以上になると演算を打切る。もしインデックスレジスタが指定されているときは番地 ADR は $(SNT)_{8\sim11}$ とインデックス・レジスタレ i 内容の和となる。指定がなければ ADR は $(SNT)_{8\sim11}$ に一致する。SNT は文字で記憶されているので数値に変換しなければならない。$(SNT)_{8\sim11}$ についてはサブプログラム ADDR で，Ri についてはサブプログラム INDX で数値に変えている。ついで操作コードを調べ，それぞれの実行プログラムに移る。もし正しい命令コードでない時は，エラー・メッセージを印刷して演算を打切る。

PLDA，PSTO の流れ図は単純であり Fig.10 に示される通りである。PADD，PSUB，PMPY，PDIV では次の 2 点について配慮している。一つは，TOSBAC－40 の整数範囲は＋32767～－32768 であるので，この範囲を超えないようにする。実際は，実数であら

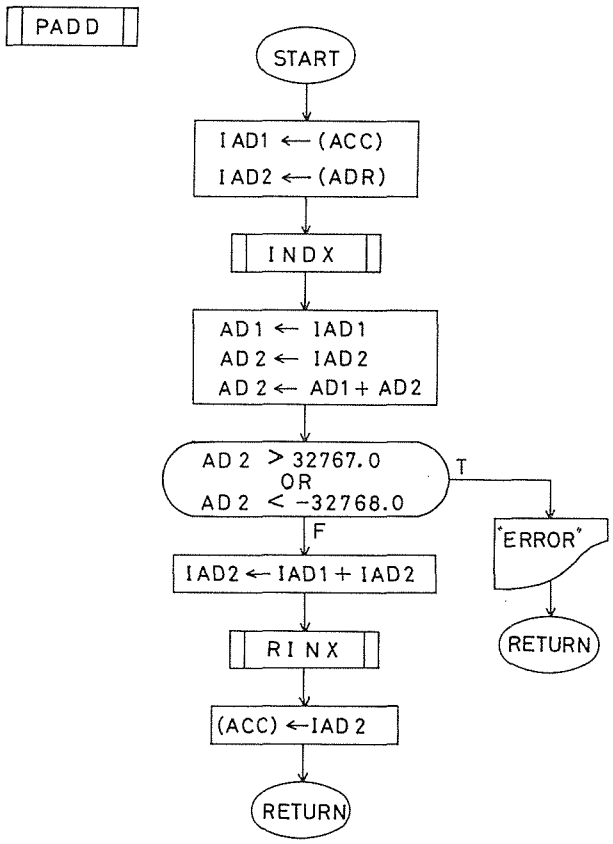


Fig. 11 Flowchart of PADD

かじめ演算してチェックをしている。これはオーバーフローなどにより制御がモニタに移らないようにするために是非必要な処置である。もう一点は，数値で演算したあとで，文字に変換してから記憶装置にもどすことである。代表して PADD について Fig.11 で説明しよう。INDX で ACC の内容および ADR の内容を整数に変換し，さらに TOSBAC－40 の機能を利用して整数を実数に変換してあらかじめ演算を行ない整数の範囲内であることを確認する。確認後，整数の演算を行ない，RINX で文字に変換して ACC に代入する。PHLT は非常に簡単であり，Fig. 8 の START の直後に制御を移せば良い。

PRWD，PWWD の流れ図を Fig.12 に示す。シフト命令の流れ図は簡単であるので省略する。

インデックス・レジスタ関係では PTIi のみを Fig.13 に示す。INDX は文字を数値に変換するプログラム，RINX は数値を文字に変換するプログラムである。ADR は分岐先の番地を表わす。必要なサブプログラムでは INDX のみを示しておく（Fig. 14)。番号2,4を付した記号はループを表わし，その内容は，初期値，増分，判断を表わす。また SUJI は0，1，2,……，9の数字を表わす。RINX は，この逆の動作をすれば良い。

他に浮動小数点（定数）の四則演算も可能であるが，ここでは省略する。

インタプリタの大きさは27キロバイトである。

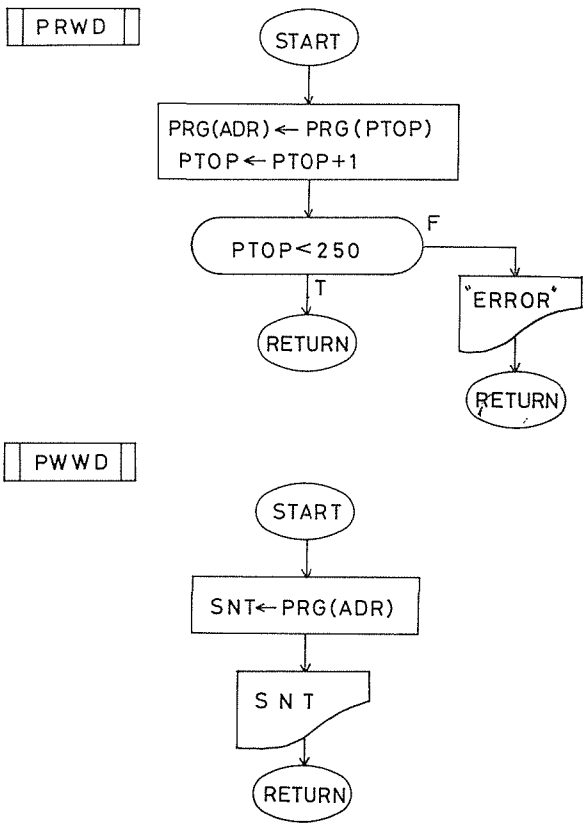


Fig. 12　Flowcharts of PRWD and PWWD.

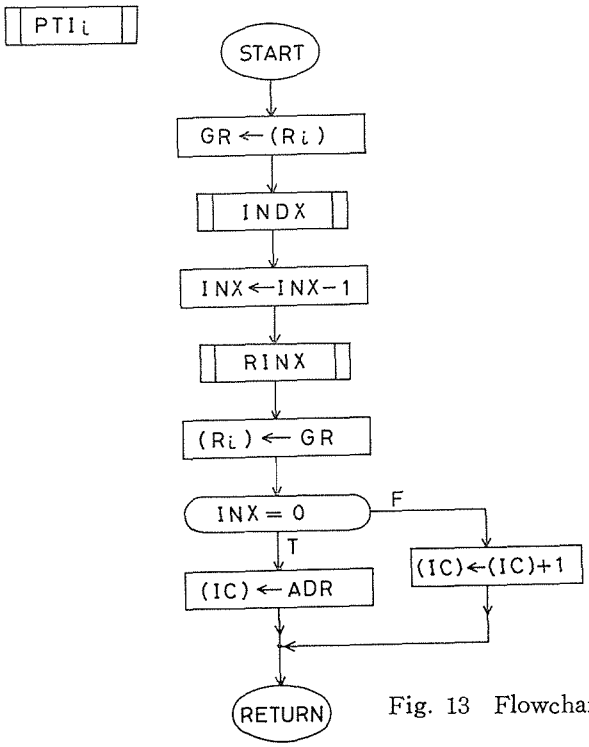


Fig. 13 Flowchart of PTIi

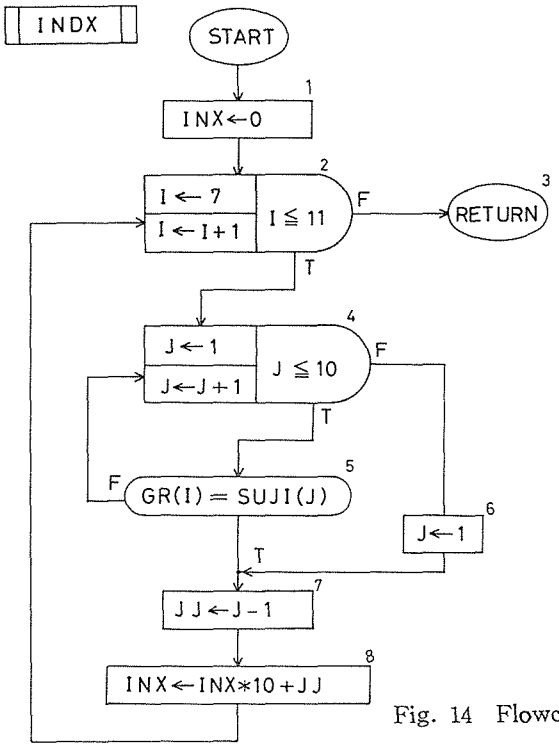


Fig. 14 Flowchart of INDX

5. SAMOSシミュレータの特長

SAMOSシミュレータの特長は，エラーの処置が完全に行われていることにあるが，他に連続処理，統計処理，デバックがある。SAMOSプログラムの例をFig.15に示す。

```
                SAMOS SIMULATION

          E8888      SAMOS
                   0  RWD0000100
                   1  WWD0000100
                   2  LDA0000100
                   3  ADD0000011
                   4  MPY0000100
                   5  ADD0000011
                   6  MPY0000100
                   7  ADD0000012
                   8  STO0000101
                   9  WWD0000101
                  10  HLT
                  11  0000000003
                  12  0000000001
                      DATA
                      0000000006
                      EOS
     0000000006
0000000343

     HALT          TIME    11
```

Fig. 15　An example of SAMOS programming.

5.1　連続処理

制御カードはSAMOS，DATA，EOSの3種であり，最初の3字SAM，DAT，EOSが予約語となっている。SAMはSAMOSのジョブの始まりを示し，DATAはデータ・カードが続くことを意味し，EOSはジョブの終了を示す。SAMおよびEOSのカードが無いとジョブは実行されない。制御文に誤りがないとき，および正常またはエラーで計算が終了したときに必要なメッセージを印刷した後で，次のジョブを読み込み実行することになる。

5.2　統計処理

SAMOSシミュレータを入門者の教育用として用いるとき，ジョブ件数，エラー件数，エラーの種類の統計が必要となってくる。Fig. 8でIDが負の時に，演算の処理が終了し，ジョブ件数，エラーの件数などが印刷される。

エラー・ナンバーの種類はTable 2の通りである。エラー・ナンバー8，9は未使用である。

Table 2 Error No.

Error No.	Meaning
1	No SAMOS Card
2	No EOS Card
3	Invalid instruction.
4	Address is outside of the range.
5	Integer is overfow.
6	Exponent of real number is overflow.
7	Invalid index.
8	Undefined.
9	Undefined.
10	Over time.

5.3 デバック

制御カード SAMOS に DB の指定があると，デバックのためのトレースを行なう。現在実行中の命令が格納されている番地，ACC の内容，命令，インデックス・レジスタの

```
                SAMOS SIMULATION

          E8888      SAMOS    'D
                 0   RWD0000100
                 1   WWD0000100
                 2   LDA0000100
                 3   ADD0000011
                 4   MPY0000100
                 5   ADD0000011
                 6   MPY0000100
                 7   ADD0000012
                 8   STO0000101
                 9   WWD0000101
                10   HLT
                11   0000000003
                12   0000000001
                     DATA
                     0000000006
                     EOS
                       0        0000000343      RWD0000100
                       1        0000000343      WWD0000100
     0000000006
                       2        0000000343      LDA0000100
                       3        0000000006      ADD0000011
                       4        0000000009      MPY0000100
                       5        0000000054      ADD0000011
                       6        0000000057      MPY0000100
                       7        0000000342      ADD0000012
                       8        0000000343      STO0000101
                       9        0000000343      WWD0000101
          0000000343
                      10        0000000343      HLT


          HALT           TIME    11
```

Fig. 16 An example of debugging

内容を印刷する。ただし印刷行数が多くなり過ぎるおそれがあるので，トレースに限り200行の印刷でジョブを終了することにしてある。トレースの例をFig.16に示す。ここではインデックス・レジスタを使用してないので，インデックス・レジスタの内容は印刷されてない。

6．エラーの解析

アセンブリ言語[8]によるプログラム作製においては連続処理のためにOSアセンブラ[9]を用いる他はなく，OSアセンブラでは入出力の指定が難かしいので，入出力はFORTRANで行なうという変則的なことをしなければならない。入出力機器を直接駆動するためにはベーシック・アセンブラ[10]を用い，ステータスを調べての制御，割込みによる制御が可能である。しかしベーシック・アセンブラでは，単位プログラムごとにオブジェクト・テープをパンチする必要があり，最小のプログラムでも20～30分の処理時間がかかり，多くのプログラムの処理には不向きである。

SAMOSではステータスによる制御，割込みによる制御などのやや高レベルの操作はできないが，一般的な機械語の知識の修得には適しており，アセンブリ言語によるプログラムよりもSAMOSによるプログラムの方が規模も大きく，かつより高級なプログラムが作られる傾向がある。入門者によるSAMOSプログラムのエラーの割合はTable 3に示す通りである。最もエラーの割合が大きいのは，操作コードの誤記であり，特にLDAをLADと書く誤りが大きい。日本語のローマ字表記では，ほとんどの子音に母音が続くことと関連のある誤りと思われ，入門者にとってLDAよりLADがより自然に見えるためのようである。次に多いのが時間超過であり，大部分は無限ループに入るか，あまりにもループが大きすぎる場合が多い。三番目に多いのが整数が大きすぎるエラーであり，TOSBAC－40では整数範囲が＋32767～－32768であり，SAMOSでは一応10桁の整数が使えるようになっているための誤りであり，本シミュレータでも将来10桁の整数を使えるように準備中である。ついで，SAMOS，EOSの制御カードのつけ忘れが多いが，これは単な

Table 3　Error rate

Error No.	Rate　(%)
1	8.3
2	6.1
3	32.9
4	4.4
5	20.6
6	0
7	0.9
10	26.8

る不注意によるものである。エラー番号6，7が少ないのは，実数やインデックスの利用が少ないためである。インデックスはループのカウンタや，番地修飾やサブプログラムの利用に欠かせない命令であるので，インデックスの使用回数が少ないのは残念である。

7. むすび

SAMOSは命令数が少く，かつ覚え易い形式であるので入門者の機械語命令の修得に適している。SAMOSの規模から考えてインデックスレジスタが3こあるのは，やや多すぎる感がする。最適なSAMOS命令の設計の基礎として最小命令構成について検討し，本論文では最小命令数が4であるとの結論を得たが，この結果より最適なSAMOS命令を構成するには，さらに多くの検討を加えねばならない。

SAMOSシミュレータはFORTRANで記述してあるので，他の電子計算機に移植することが容易であり，マイクロコンピュータに移植できれば，入門者の学習用として理想的であろう。特にデバック機能を用い，デイスプレイで表示すればデバックが一段と使いやすくなるであろう。

SAMOSシミュレータは，まだ不完全な部分があり，さらに改訂を要する。終りに，協力していただいた山形大学計算センター職員に感謝の意を表す。また本論文の一部を講演発表[11]したことを付記する。

参考文献

1） 中津山，池田：“小形電子計算機に向いたアセンブラ・レベルの構造化プログラミング指向言語”，山形大学紀要（工学），16，1，147～158（1980）

2） Forsythe, A. I., Keenan, T. A., Organick, E. I. and Stenberg, W. : COMPUTER SCIENCE A FIRST COURSE, JohnWiley & Sons, Inc., New York (1969)

3） 渡辺：“4記号6状態の万能 Turing 機械”，情報処理，13，9，588－592（1972）

4） McCarthy, J., Abrahams, P. W., Edwards, D. J., Hart, T. P. and Levin, M. I : LISP 1.5 Programmer'S Manual, The M. I. T. Press (1965)

5） Kemeny, J. G. and Kurtz, T. E. : BASIC Programming, John Wiley & Sons, Inc., New York (1967)

6） Mckeeman, W. M., Horning, J. J. and Wortman, D. B. : A compiler generator, Prentice-Hall, Inc. (1970)

7） Donovan, J. J. : System Programming, McGraw-Hill. Inc. (1972)

8） 浦編：アセンブリ言語，培風館（1970）

9） ソフトウエア技術部：TOSBAC－40プログラミング説明書OSアセンブル編，東芝（株）（1971）

10） 産業用電算機部：TOSBAC－40プログラミング説明書アセンブラ編，東芝（株）（1971）

11） 中津山：“機械語教育用SAMOSシミュレータ”，電気関係学会東北支部連大，2E14（1979）

山形大学紀要（工学）第17巻　第1号　昭和57年1月

SAMOS Simulator and its Minimum Set of Instructions

Mikio NAKATSUYAMA, Mitsuru MIZUNUMA, Minako SUGIMOTO, Norio NISHIZUKA and Hiroshi NAGAHASHI

Department of Electronic Engineering, Faculty of Engineering

The machine language SAMOS which consists of the simplest Instructions is suitable for the education of a machine language. We designed the SAMOS simulator which is implemented on a mini-computer TOSBAC-40. The SAMOS simulator is a kind of an interpreter written by FORTRAN. Even for untrained programmers, the substantially larger program can be performed by SAMOS than by a conventional assembly language.

The problems of the minimum set of instructions have been discussed on Turing machine. We tried to obtain the minimum set of instructions of SAMOS and got the following result: Four instructions are sufficient to produce most programs. The concluding remarks about the minimum set of instructions are useful for the design of an efficient machine language, although minimum set of instructions are not always convenient for programmers.